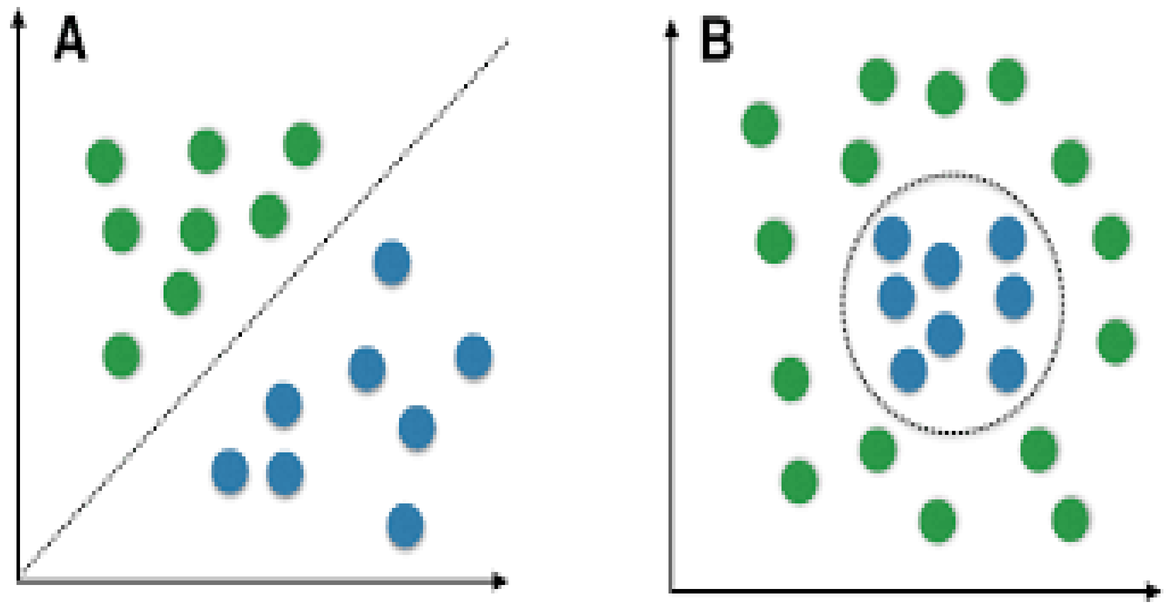
NAÏVE BAYES CLASSIFIER

Wande B. Adeyeye

PROJECT REPORT 2020



SPAM EMAILS CLASIFICATION

BY:

WANDE B. ADEYEYE

E-MAIL:

ADEYEYEWANDE@GMAIL.COM

The very first step in any Data Science and Machine Learning problem solving is asking questions and I mean asking the **right questions,** as a Data Scientist you must be capable of asking the right questions because if we get the question wrong i.e. if we asked the wrong questions then we will get our algorithm perfectly wrong because the questions we asked will determine how we will go about writing the algorithm to solve it.

FIRST STEP (Forming The Questions):

In this project we want to filter out spam emails and classify our emails into spam and non-spam, the first step is to identify the problems and form our questions we're going to solve from the problem, by doing that simply means we turn business problem into machine learning problem and what we mean by that is we have clear business objective here, we receive the emails from a source and have to filter out the spam emails. What it means from machine learning stand point is we first have to find which emails are spam and which emails are legitimate i.e. for each email we have to assign a category if the email is spam as spam or legitimate as non-spam by our algorithm. So, our objective is to learn what characterize a spam email so we can start to classify all the emails. This makes our project a classification problem because here we're separating the data unlike regression where we fit all the data together. In summary here is our objectives:

- Take the raw emails, pre-process the text data
- Train a machine learning model that classifies it as either spam or non-spam
- Test and evaluate the performance of our trained model

SECOND STEP (Gathering the Useful Data):

After we've successfully nailed our questions, the next thing we're going to do is gather useful data that will help us answer the questions, the data we'll be working with comes from an anti-spam open source platform called spam assassin it has huge spam and legitimate emails they made available to the public and with constant update and the link to the files are here or direct search from google here.

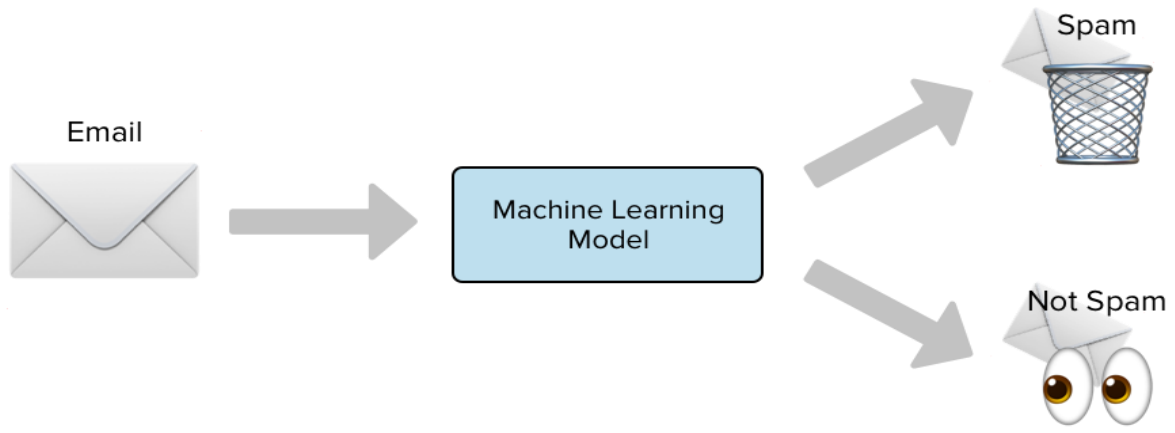
NOTE:

Corpus is a collection of written texts, especially the entire works of a particular author or a body of writing on a particular subject. For example, "the Darwinian corpus" whereas a Document refers to as a piece of item in our corpus. The plural for corpus is "corpora". These two words will come up again and again in this module that's why I put it here for reference.

Now that we’ve formed our questions and as well gathered our data, I want to talk briefly about what we’re going to do in this project, the theory behind the machine learning model we’re going to use and the machine learning were going to use in this module is called the Naïve Bayes classifier, it’s as a result of its simplicity and speed given that were looking to classify spam emails, speed is a really important attribute to have, the speed of Naïve Bayes has made it one of most popular machine learning model in spam classification and is heavily used today, the model comes from its simplicity and the simplicity of this machine learning model will help us build our own model in a relatively short amount of time.

How Naïve Bayes Classifier Works:

To make a decision whether an email is spam or not, what the Naïve Bayes classifier does is it compare two probabilities, the chances, the likelihood of an event happening, what our algorithm will do is it will calculate the probability of an email being spam and the probability of an email being legitimate so if the probability of an email being spam is higher, then our algorithm will classifier such email as spam basically our algorithm will look at two numbers and classifier based on which is higher.



We can do this by using a simple, yet powerful theorem from probability theory called Bayes Theorem. It is mathematically expressed as:

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)},$$

where A and B are events and $P(B) \neq 0$.

- $P(A)$ and $P(B)$ are the probabilities of observing A and B without regard to each other.
- $P(A \mid B)$, a conditional probability, is the probability of observing event A given that B is true.
- $P(B \mid A)$ is the probability of observing event B given that A is true.

Problem Statement

We have a message m = (w1, w2, ... , wn), where (w1, w2, ... , wn) is a set of unique words contained in the message. We need to find

$$P(spam|w1 \cap w2 \cap ... \cap wn) = \frac{P(w1 \cap w2 \cap ... \cap wn|spam).P(spam)}{P(w1 \cap w2 \cap ... \cap wn)}$$

If we assume that occurrence of a word are independent of all other words, we can simplify the above expression to

$$\frac{P(w1|spam).P(w2|spam)...P(wn|spam).P(spam)}{P(w1).P(w2)..P(wn)}$$

In order to classify we have to determine which is greater

$$P(spam|w1 \cap w2 \cap ... \cap wn) \; versus \; P(\sim spam|w1 \cap w2 \cap ... \cap wn)$$

Third Step (Data Cleaning):

We import all the needed libraries and files however I started by reading a single file to set as an example of how the rest of the files will be handled. After we've import our date in this case the emails, the next thing we are going to do is to extract the text in the email body from all of our emails, we do that by creating a generator function that will hold onto all of the spam emails using pandas data frame function and we as well do the same for the non-spam (ham) emails after we've done that, we will group all the emails together both the spam and the non-spam using pandas concat function.

At this point we have:

- Extract relevant data (Email bodies)
- Convert from text files to DataFrame

The next step is to:

- Check for empty emails
- Check for null or missing values

Missing Value Detection: Missing data pattern was used to identify the missing data in the dataset. It can be observed that the data does not consist of any missing data

Summary of other Data Inconsistencies: While exploring the data we found an instance where the data was inconsistent and didn't make logical sense. We choose to make the values consistent by excluding it from the data.

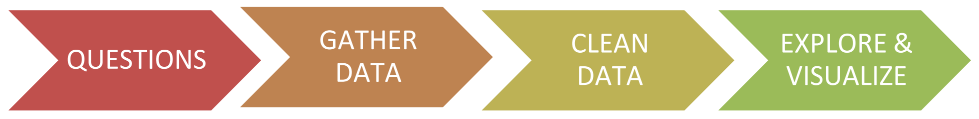
- Two observations with empty emails was located

So now we have:

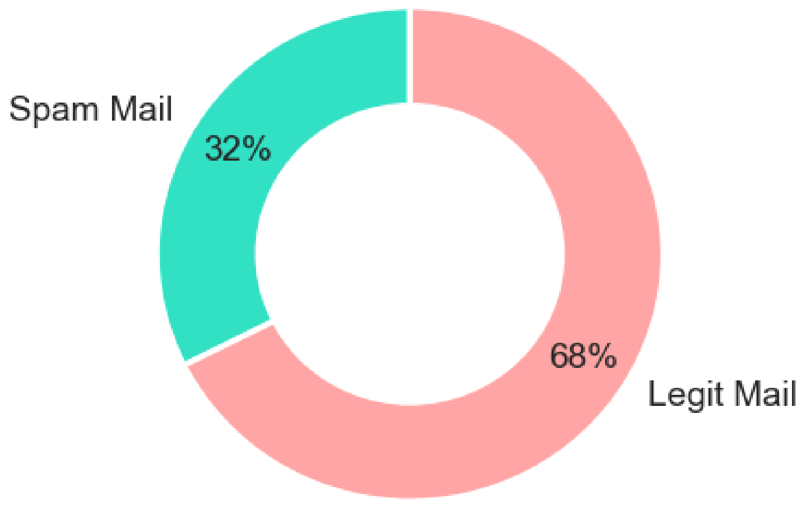
- Extract relevant data
- Convert from text files to DataFrame
- Checked for empty emails
- Checked for null and missing values
- Removed rows with bad data for our DataFrame

What we do next is to save our file as JSON format into our machine. The next step is to explore and visualize out data.

Fourth Step (Explore & Visualize):



Since we are working categorical data with just two categories, we will visualize using a donut chart to see the percentages of both categories. Pie chart is commonly used today especially in business reports, so it's as well important to get this chart under our belt. Looking at the visualization below, we can see the percentage of both category each assume in our data.



Natural Language Processing (NLP)

Our next step and what we will be talking about for alittle is natural language processing (NLP), how we're going to use NLP for our naïve bayes classifier is we'll use it to prepare a piece of text for our learning algorithm, we have to convert our email body to a form that the algorithm can understand which means preprocessing our text and by preprocessing simply means:

- Tokenizing
- Converting our text into lower case
- Removing stop words
- Word Stemming
- Removing puntuations
- Stripping out the HTML Tags

To active all this, we need to make use of python package called The Natural Language Toolkit or NLTK and for more the documentation is here. This is a package that every NLP professional will use at some point in their field. The NLTK can do a lot of things and we're going to start using them for our fundementals that is preprocessing our text so that our machine can use it. We starts by importing the libraries and download the NLTK resources, we download the tokenizer and the stopwords. The tokenizer divides a text into a list of sentences whereas the Stopwords are the words which does not add much meaning to our text such as 'it', 'the', 'is' and so on. After tokenizen and removed the stopwords our next step is to stem the words i.e Word Stemming, and remove puntuations, Stemming is the process of reducing a word to its word stem that affixes to suffixes and prefixes or to the roots of words known as a lemma. The ideal behind word stemming is to treat inflected and derived words in the same way. Next is removing the HTML tags, HTML tags are the hidden keywords within a web page that define how your web browser must format and display the content.

So now we are done with:

- Tokenizing
- Converting our text into lower case
- Removing stop words
- Word Stemming
- Removing puntuations
- Stripping out the HTML Tags

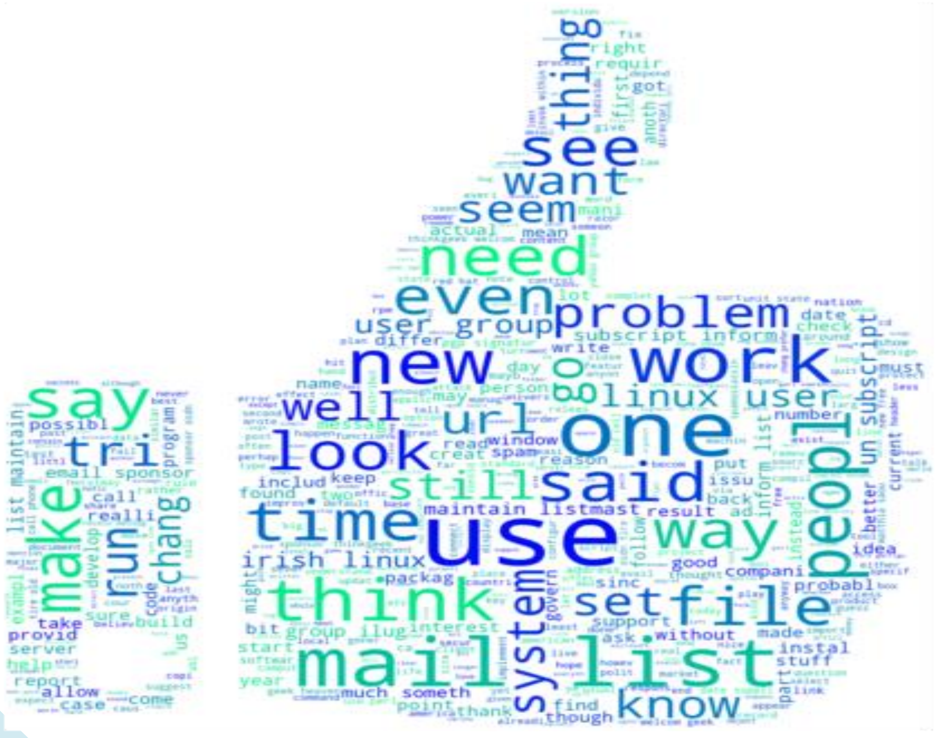
What we're going to do next is this, we're going to create a function to hold on to our clean email bodies and return filtered words and assigned them to a nested list.

Word Cloud Visulization

A word cloud (tag cloud or wordle or weighted list in visual design) is a novelty visual representation of text data, typically used to depict keyword metadata (tags) on websites, or to visualize free form text. Tags are usually single words, and the importance of each tag is shown with font size or color. An example is show below:

We'll create a word cloud each for both the spam and the non-spam emails, using the Thumbs up mask for the non-spam and Thumbs down mask for the spam emails as shown below:

Non-Spam (Legitimate) Email below:



Spam Emails below:

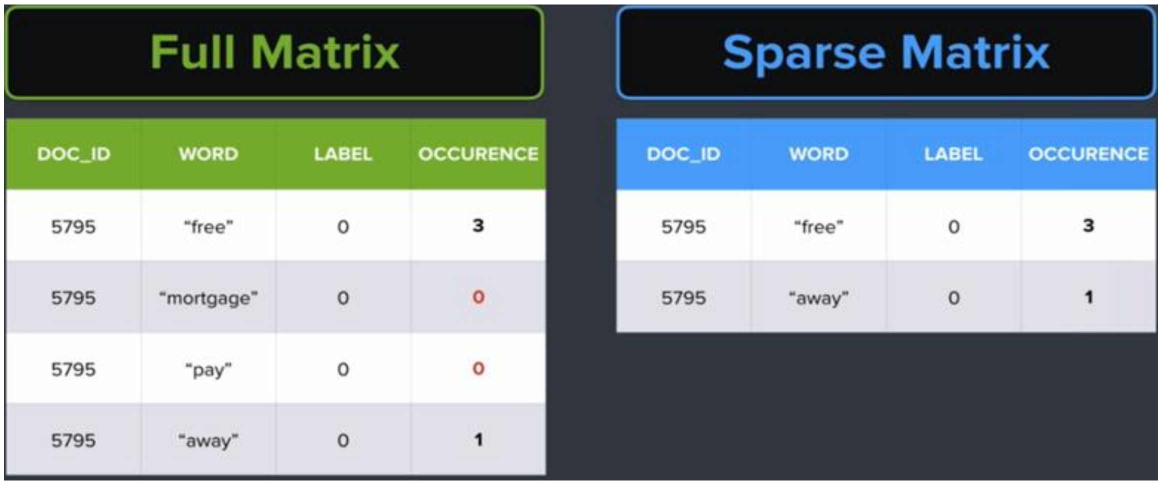
REMOV MAIL
SYSTEM
LOOK
INTEREST
MAKE
WEB SITE
FREE
BUSI
GIVE
THANK
WANT
PERSON
COMPANI
CREDIT CARD
MAY
BROADCAST EMAIL
LIST PLEAS
EMAIL ADDRESS
PEOPL
ORDER REPORT
WISH RECEIV
PRODUCT
USE
ONE
HELP
GO
MAKE MONEY
CLICK LINK
MESSAG MIME
MIME FORMAT
RECEIV EMAIL
MAIL LIST
PAY
IMMEDI
UNIT STATE
WORK
SEE
TIME
SERVIC
FILL FORM
INTERNET
NEED
RECEIV MAIL
SITE
OFFER
PROVID
PROGRAM
MARKET
YEAR
REPLI MESSAG
ACCOUNT

<u>**Create vocabulary for Spam Classifier**</u>

We go back to preprocessing our data for our bayes classifier, there are lots of unique words in our emails that constitute our data, we're not going the use the total of the 5847 words instead we are going to use the 2500 of the most frequent words. The 2500 most frequent words in our data are going to generate our vocabulary and we will generate our vocabulary from our stemmy list of words. To do this, we call our stemmed clean message with no HTML tag, we assigned it to stemmed nexted list and flatted it, to get our unique words we create a pandas dataframe and called the value count function to get the unique words values and as shown, we have 2703 unique words in our email bodies looking at this figure it's a specific huge number and we are only going to train our classifer with a subset of this number that is 2500 of this words. We assign 2500 to the VOCAB_SIZE, covert our frequient words to list and then create a pandas dataframe for the uniques words and save tham as csv files in our machine.

<u>**Features as a Sparse Metrix**</u>

The final steps of the data preprocessing for out bayes classifer is creating our features as a sparse metrix, its important to use it because were only using the VOCAB_SIZE of our email. The difference between the **Full Metrix** and **Sparse Metrix** is: **Sparse matrices** provide efficient storage of double or logical data that has a large percentage of zeros. While **full (or dense) matrices** store every single element in memory regardless of value, sparse matrices store only the **nonzero** elements and their row indices. And that means we only includes the rows which have a word that occur in the mail.



Full Matrix

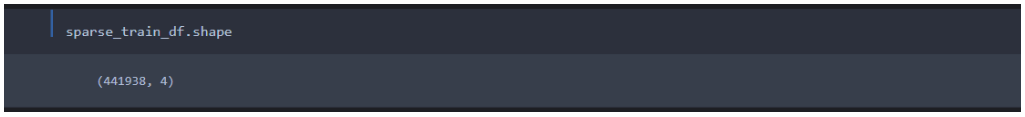
DOC_ID	WORD	LABEL	OCCURENCE
5795	"free"	0	3
5795	"mortgage"	0	0
5795	"pay"	0	0
5795	"away"	0	1

Sparse Matrix

DOC_ID	WORD	LABEL	OCCURENCE
5795	"free"	0	3
5795	"away"	0	1

Looking at the example above the words 'mortage' and 'pay' has zero occurrence, it was present in the full metrix but was excluded in the sparse matrix. It's just a compression of the full matrix with a fewer rows.

We start by converting the 'stemmed_nested_list' which is a pandas series to list and create a pandas dataframe with one word per column. Next step is splitting the data into training and testing dataset using scikit learn. After we've successfully split our data into train and test dataset, we will create sparse matrix for all of the trainning dataset which in total includes '441938' rows,

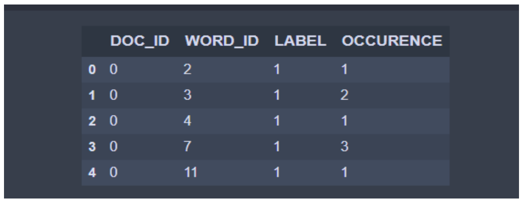


```
sparse_train_df.shape

    (441938, 4)
```

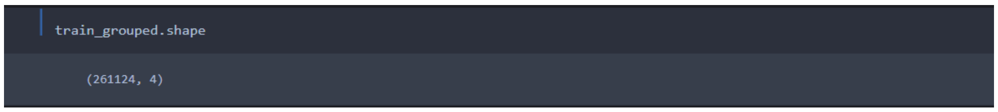
we have this huge number because we've put each and every single words from trainning dataset into a separate row, so if in the same we have a word occur twice, both will be put in two separate rows in the data, what we are going to do now is to combine this occurrences that is if a word occur more than one time in an email, we should conbine it in the state of the frame.

We are going to group all word by email using the pandas groupby() method. We will group it by document ID, word and label then we sum up the number of occurrence. From the table below, for document ID zero we have word ID that are grouped together, the word with id 7 occur three times in the first email.



	DOC_ID	WORD_ID	LABEL	OCCURENCE
0	0	2	1	1
1	0	3	1	2
2	0	4	1	1
3	0	7	1	3
4	0	11	1	1

After we've succefully grouped our trained emails using the pandas groupby() method, the size of our dataframe has significantly reduced, it has gone from 441938 to 261124 rows as seen below, and this put us in a good place to save our work and we do that by creating a path and save it as a text file.



```
train_grouped.shape

    (261124, 4)
```

We will do the same for the test dataset and save it as a text file in our machine. After we complete this process successfully, we’ve come to the end of data preprocessing and from here we start trainning our naïve bayes classifier to create our spam filter.

<u>Train Naïve Bayes Classifer</u>

We start by importing the needed libraries, constants and then load the text files we saved from previous steps in preproccessing using numpy load text method both train and test text files. Using pandas dataframe function to print the head of our text files to see the top of our dataset, we run some EDA to check and set up our notebook. The format of our dataset is in four colum, the first one is a Document I.D. which identifies a particular email, the second is Word I.D. which identifies a token or word, the third is our label or category that is zero ‘0’ for non-spam and one ‘1’ for spam, and the forth one is Occurrence, that is the number of times a particular word occur in the email. Next, we’ll transform the numpy array to numpy dataframe, we are going to restructure it from a sparse matrix to full matrix. We do that by creating an empty dataframe and populate it with values, we create a dataframe first with the features needed after we defined a function to populate the dataframe to transform from sparse matrix to full matrix, it will populate the empty dataframe we created with the VOCAB_SIZE which is 2500

QUESTIONS
GATHER DATA
CLEAN DATA
EXPLORE & VISUALIZE
MODEL

We will start training our naïve bayes model by calculating the probability of each individual tokens, here we will work out the probability for each token. We discuss the probability behind this model earlier in this project which is bayesian theorem and we can always refer back to it. After succefully calculations, here are the numbers:

- Probability of an email been spam is approximately 31%
- Total number of words is approximatley 441938
- Number of non-spam or ham words are 258293
- Number of spam words are 183645
- Spam emails have a higher average words in a single mail

Our next step is to sum up the tokens that occur in the spam and ham emails and we need to do this for each word ID for both the spam and ham emails respectivly. We sum all the tokens column by column. That is all the column under the number zero will be summed up as well as number one and so on across the Word ID, what we will end up with is a panda series with all the Word ID with the number of times these tokens occur in spam messages. We will do the same for the ham messages and saved to ‘summed_spam_tokens’ and ‘summed_ham_tokens’ respectively. Next we will calculate the probability that a token occur given that the email is spam that is conditional probability, also the probablility that a token occur given the email is non-spam, we will do this for all our token simultaneously and then we calculate the probability that a token occurs reagradless whether it’s spam or non-spam emails. After we’ve succefully done that, we will save them by first creating a path and then using the numpy save text function to save it in our marchine that is:

- Spam probability token
- Ham probability token
- All the probability token

Before we move on we can simply prepare our test text files we will be using later on as we continue, we can call the function we wrote before ‘make_full_matrix’ and run our test through it and save it to our marchine

- Test target files
- Test features files

At this point we’ve done a lot of work and we’re in the position to run our algorithm and make some predictions.

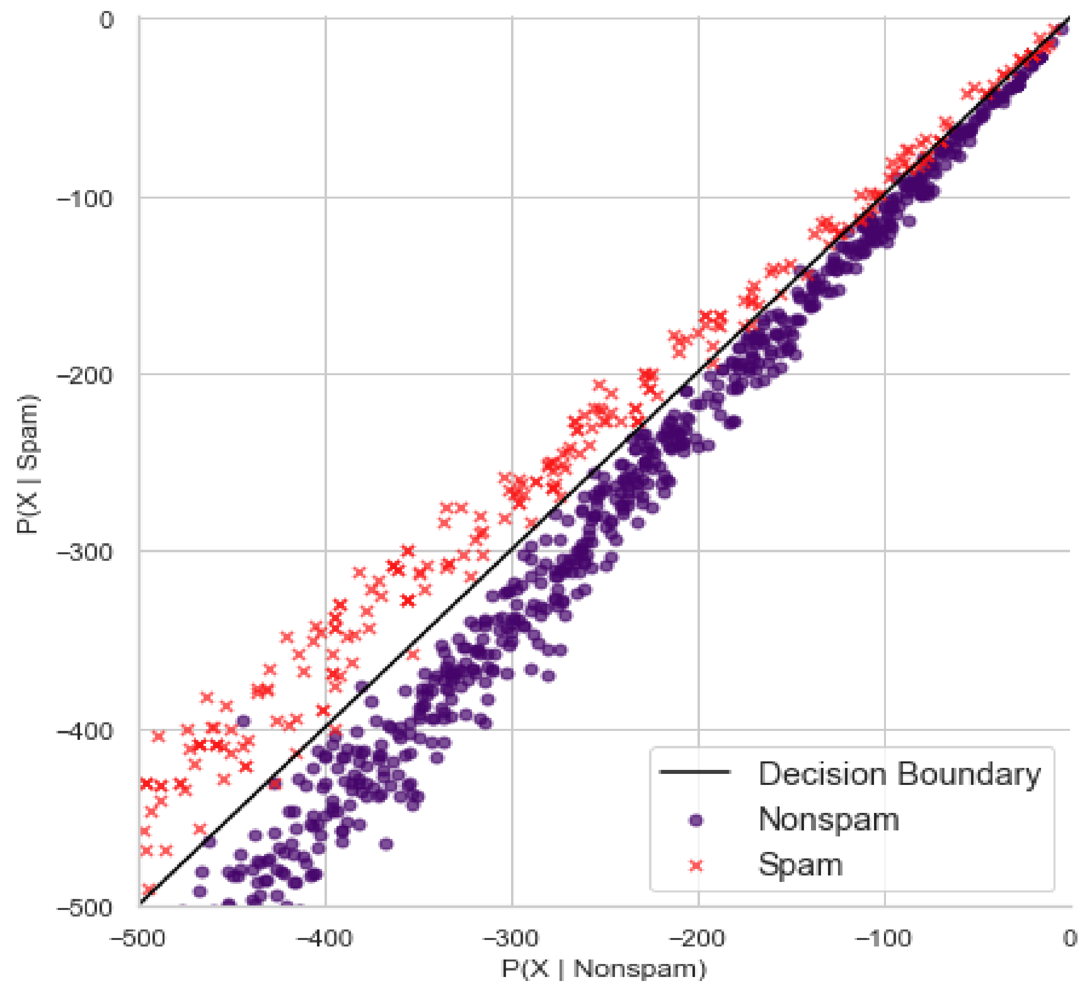
It’s advisible and I save the data to text files, we do this because to continue will not require us to run all our lines of code and the number and values remain the same. For testing, inferance and evaluation, we will start from a fresh notebook, we import needed libraries and the text files we created from our previous notebook and load our text data i.e features, target and token probabilities. What we are going to do next is to put all the pieces together i.e conditional probability, joint probability, independence assumption and bayes theorem, we’ve already talked about the probabilities at the top of the report and we can always refer back for a quick recap. We’ve worked out this numbers and tokens, we have saved them to our text files that we’ve imported into our notebook, what we are yet to do is to mulitiply all the values together for all the tokens and that’s what we are going to do now by using numpy dot product for the ‘X_test’ and ‘prob_token_spam’ next let’s talk about the prior, it simply means the probability of an event will be revised as new data or information becomes available, to produce a more accurate measure of a potential outcome. That revised probability becomes the posterior probability and is calculated using **Bayes' theorem**. We will calculate our probability in log format, we are going to combine joint and conditional probability and calculate the probability an email is spam given tokens and we will do this for all of the emails in our test dataset and assigned it to ‘joint_log_spam’ and we’ll do the same for our non-spam emails as well.

We will be making prediction based on the two joined probabilities that we previously calculated, we will be comparing the two and classify emails based on the one with higher probability, we create our y hat, with true is equals to ‘1’ and false is equals to ‘0’, our next step is to evaluate our model.

We do this in order to see how well our model is doing, we will check how many emails our model classifier correctly, we do that by calculating the accuracy, and from the result our model did great out of over 1700 emails it misclassifier only 44 of them, next we will quantify the model accuracy.

- 97% accuracy

Next let's visualize our result with some scatter plot that will show us all that is happening behine the scenes. The chart below sheard more light on how our model classify the emails in between we have the decision boundry that sperate the spam emails from the non-spam emails but if we look closely we see some instant in which some legit (non-spam) emails overlap to the spam emails class and vice versa that clearly shows the calculation we did earlier and show that our model is not 100% accurate non the less it's just perfect



False Positive Vs. False Negative

A false positive means that the results say you have the condition you were tested for, but you really don't. With a false negative, the results say you don't have a condition, but you really do. We will be talking about the decision our model get right and the decition it got wrong, we've previously look at the accuracy on how well our model was right but here sometimes some spam emails do make it to our message inbox, that's known as false negative and as well some legit emails end up in spam folder that is false positive. In some cases we had to go in the spam folder to get our actual email, so using the numpy unique function, we can calculate the number of false positive and false negative in our model. Out of all of our test emails, here are the numbers:

- 13 False Positive
- 31 False Negative

The next metric we are going to look at are Recall Metric, Precision Metric and F-score or F1 Metric.

Recall Metric is the ability of a model to find all the relevant cases within a dataset. The precise definition of recall is the number of true positives divided by the number of true positives plus the number of false negatives.

Precision is a metric that quantifies the number of correct positive predictions made. Precision, therefore, calculates the accuracy for the minority class. It is calculated as the ratio of correctly predicted positive examples divided by the total number of positive examples that were predicted

F-score, also called the **F1-score**, is a measure of a model's accuracy on a dataset. The F-score is a way of combining the precision and recall of the model, and it is defined as the harmonic mean of the model's precision and recall.

After running our code we come out with the following output **repectively**:

- 95%
- 98%
- 96%

Now to our final step we will be implementing our naïve bayes classifier using SciKit Learn model we will start from a fresh notebook as well and we will be using the Json file we saved during the data pre-processing stage, that's the data file we will be using. First we will import our libarary, and we add a string to the path of the json file we will be using, after that we assigned the json file to data and open the file using pandas read function next we will drive straight into generating our vocabulary for our bayes classifier using sk learn features extraction then we will create our vectorizer and use english stop words to remove all the stopwords in all our emails. Now we can create our vocabulary and document term matrix, next we using all the features, vectorizer and fit it with the message column from our data frame at the end we get the sparse matrix, the column correspond to each token, individual words in our emails. Now that we've get our feature matrix and vocabulary words next we are going to split our data into train and test dataset and we supply our features and labels after that we are going to train our model MultinomiaNB which will allow us to create our models quickly, we assigned it to classifier and call the fit method and supply our X train and y train in parentesis. To get a look on how well our model does, we will do some quick calculations, we want to know how well the model classified our emails correctly

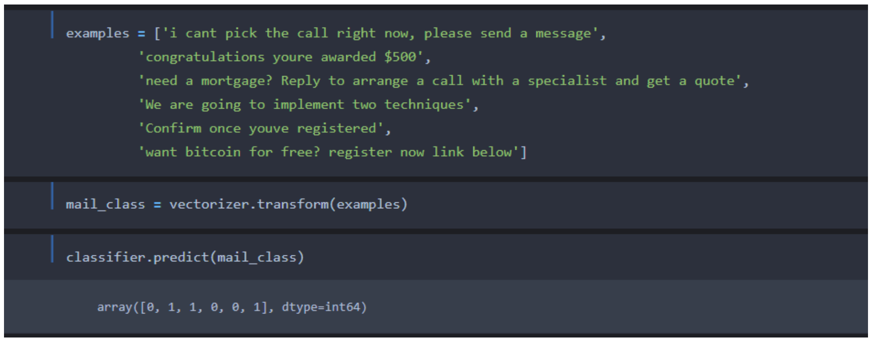
- 1382 documents classfied correctly
- 80 documents classfied incorrectly
- The (testing) accuracy of the model is 95%

Lets go beyound accuracy and to look at are Recall Metric, Precision Metric and F-score or F1 Metric. For our classifier

After running our code we come out with the following output **repectively**:

- 83%
- 99%
- 91%

Here are the output and they are looking very strong. Since we've trainned our classifier, we can evaluate some sentences or emails to see how our model classifies it. Looking at the output we can confidently say our model did well in classifing the emails



```
examples = ['i cant pick the call right now, please send a message',
           'congratulations youre awarded $500',
           'need a mortgage? Reply to arrange a call with a specialist and get a quote',
           'We are going to implement two techniques',
           'Confirm once youve registered',
           'want bitcoin for free? register now link below']
```

```
mail_class = vectorizer.transform(examples)
```

```
classifier.predict(mail_class)
```

```
    array([0, 1, 1, 0, 0, 1], dtype=int64)
```

The objective and goal of this project from scratch is to:

- Take the raw emails, pre-process the text data
- Train a machine learning model that classifies it as either spam or non-spam
- Test and evaluate the performance of our trained model

And that's exactly what we have done.

Useful links:

Dataset link [here]

Bayes theorem link [here]

Word cloud documentaton [here] and [here]

Fonts from google [here]

Masks used and others link [here]

My Github profile [here]